Implementasi Load Balancing Web Server Menggunakan Metode LVS-NAT

Jefry Alvonsius Rabu[1]
jefryrabu@gmail.com

Joko Purwadi[2]
jokop@ukdw.ac.id

Willy S. Raharjo[3]
willysr@ti.ukdw.ac.id

Abstract

Web server plays a vital role on serving requests from clients. As Internet users grows so fast, the request number increases significantly, thus reducing the overall performance of the web server. One solution is to implement a load balancing system.In this paper, we implement load balancing with Linux Virtual Server to distribute the load to several machines within a cluster. We compared Round Robbin and Least Connection algorithm.

The study reveals that load balancing using LVS-NAT can double the throughput output compared to single web server, but less significant on response time and CPU utilization. Implementing LVS-NAT using round robbin algorithm is more robust in optimizing the throughput, CPU utilization, and response time compared to least connection.

Kata Kunci : Web Server, Load Balancing, LVS-NAT, Round Robbin, Least Connection

1. PENDAHULUAN

Salah satu jenis layanan dari internet yaitu *World Wide Web*. *World Wide Web* atau *web* adalah suatu cara mengakses informasi melalui media internet. *Web* bisa juga dikatakan sebagai suatu model berbagi informasi yang dibangun di atas media internet. *Web* menggunakan protokol HTTP untuk mengirimkan data. Data tersebut tersebar di seluruh penjuru dunia disimpan dalam media penyimpananan berupa *server*. *Web server* bertanggung jawab melayani permintaan HTTP dari aplikasi klien yang dikenal dengan *web browser*. *Web server* akan mencari data dari *Uniform Resource Locator* (URL) yang diminta dan mengirimkan kembali hasilnya dalam bentuk halaman-halaman *web* yang umumnya berbentuk dokumen *Hypertext Markup Language* (HTML) dan semua isi (*content*) dari suatu situs ke komputer klien.

Seiring dengan berkembangnya kebutuhan pengguna dan peningkatan permintaan pada situs maka kerja dari *web server* bertambah berat. *Web server* yang handal selayaknya mampu melayani *request* dari pengguna dalam jumlah yang cukup besar dalam satu satuan

[1] *Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana, Yogyakarta*
[2] *Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana, Yogyakarta*
[3] *Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana, Yogyakarta*

waktu. Namun terkadang *web server* mengalami *down* atau *fail* dimana *web server* tidak dapat mampu lagi menangani jumlah request yang sangat besar dalam satu satuan waktu tersebut.

Salah satu solusi yang dapat dilakukan untuk mengatasi masalah performa *web server* dalam hubungannya dengan jumlah *request* yang meningkat adalah pemutakhiran perangkat keras *web server,* namun solusi ini hanya bersifat sementara. Maka dengan ini perlu diterapkan suatu implementasi teknologi yang dapat menjadi solusi alternatif masalah di atas. Teknik yang dianjurkan pada penelitian ini adalah implementasi *load balancing* dimana beban kerja *single server* dibagi ke dalam beberapa *server* yang ada.

2. DASAR TEORI

2.1. Web Server.

Gourley,et al.(2002), dalam bukunya yang berjudul "*HTTP - The Definitive Guide*" menjelaskan bahwa *web server* dapat mengacu kepada perangkat lunak maupun perangkat keras tertentu atau komputer yang memuat halaman – halaman *web*. Sebuah *web server* bertugas memproses *HTTP request* dan memberikan respon terhadap *request* tersebut.

Menurut *Mitch Tulloch* dalam "*Microsoft Encyclopedia of Networking*" *web server* didefenisikan sebagai suatu aplikasi yang mendukung *Hypertext Transfer Protocol* (HTTP) pada sisi *server*. *Web server* digunakan untuk mem-*publish* konten pada intranet perusahaan tertentu atau melalui jaringan internet. *Web server* menyimpan konten dari sebuah *web*. *Web server* bekerja menggunakan protokol HTTP oleh karena itu sering kali disebut HTTP *server*.

2.2. Uji Performansi *Web Server.*

Uji performansi web server memiliki bentuk-bentuk tersendiri sebagai berikut:

- Performance Test.

  Uji performansi digunakan untuk menguji setiap bagian dari suatu *web server* dengan tujuan untuk menemukan teknik terbaik mencapai optimasi ketika trafik *web* meningkat.
- Load Test.

  Load test dilakukan dengan pengujian *website* menggunakan estimasi trafik dari sebuah *website* yang mampu dilayani. Caranya adalah mendefinisikan waktu maksimum sebuah halaman *web* dimuat. Pada akhir pengujian dilakukan pembandingan seberapa maksimum waktu yang dibutuhkan untuk membuka halaman *web* pada sebuah *web server*.
- Stress Test.

Stress test adalah berupa simulasi serangan "*brute force*" yang menjalankan muatan atau permintaan secara berlebihan menuju *web server*. Tujuan *stress test* adalah untuk mengestimasi muatan maksimum sebuah *web server* sanggup menanganinya.

2.3. Load Balancing.

Server load balancing adalah sebuah proses atau teknologi yang mendistribusikan trafik sebuah situs kepada beberapa server menggunakan sebuah perangkat jaringan. Perangkat tersebut menerima trafik yang ditujukan pada suatu situs dan mendistribusikannya ke beberapa server. Proses *load balancing* sepenuhnya transparan bagi *end-user*. *Load balancing* dapat diimplementasikan dengan *hardware* khusus, *software* maupun gabungan keduanya. Konfigurasi standar yang ada memberi gambaran bahwa satu mesin ditempatkan diantara *klien* dan *server*, mesin ini disebut sebagai *director* karena tugasnya adalah memberikan balancing pada *request* dari *klien* ke *server*. Sebuah *load balancer* adalah perangkat jaringan yang dipasang diantara *klien* dan *server*, bekerja sebagai saklar untuk *request* dari *klien*. *Load balancer* mengimplementasikan beberapa metode penjadwalan yang akan menentukan ke arah *server* mana *request* dari *klien* akan diteruskan. Beberapa keuntungan yang diperoleh dari teknik *load balancing* sebagai berikut:

- *Flexibility : Server* tidak lagi menjadi inti sistem dan resource utama, tetapi menjadi bagian dari banyak *server* yang membentuk *cluster*. Hal ini berarti bahwa performa per unit dari *cluster* tidak terlalu diperhitungkan, tetapi performa *cluster* secara keseluruhan. Sedangkan untuk meningkatkan performa dari *cluster*, *server* atau unit baru dapat ditambahkan tanpa mengganti unit yang lama.
- *Scalability :* Sistem tidak memerlukan desain ulang seluruh arsitektur sistem untuk mengadaptasikan sistem tersebut ketika terjadi perubahan pada komponen sistem.
- *Security :* Untuk semua trafik yang melewati *load balancer*, aturan keamanan dapat dimplementasikan dengan mudah. Dengan *private network* digunakan untuk *real servers*, alamat *IP*nya tidak akan diakses secara langsung dari luar sistem *cluster*.
- *High-availability : Load balancer* dapat mengetahui kondisi *real server* dalam sistem secara otomatis, jika terdapat *real server* yang mati maka akan dihapus dari daftar *real server*, dan jika *real server* tersebut kembali aktif maka akan dimasukkan ke dalam daftar *real server*. *Load balancer* juga dapat dikonfigurasi *redundant* dengan *load balancer* yang lain.

2.4. Linux Virtual Server.

Virtual server adalah *server* yang mempunyai skalabilitas dan ketersedian yang tinggi yang dibangun diatas sebuah *cluster* dari beberapa *real server*. Arsitektur dari sebuah

server cluster adalah benar-benar transparan sampai ke *end-user* dan masing-masing *user* berinteraksi dengan sistem seolah-olah hanya ada satu *virtual server* dengan performa yang tinggi.

Linux Virtual Server (LVS) menerapkan proses pemilihan pada layer 4 (Transport) dari *kernel* Linux. LVS meneruskan *session* TCP dan UDP untuk menyeimbangkan beban melalui beberapa *real server*. LVS berjalan pada Linux dan dapat menyeimbangkan koneksi dari *end-user* dengan sistem operasi apapun kepada *real-server* yang menjalankan sistem operasi apapun, selama koneksi yang dilakukan menggunakan TCP atau UDP. *Linux virtual server* (LVS) adalah alternatif utama *open source* yang menyediakan solusi untuk menciptakan sistem *load balancing*. Seperti yang ditulis di *home site Linux Virtual Server project*, "*Linux virtual server is a highly scalable and highly available server built on a cluster of real servers, with the load balancer running on the linux operating system*".

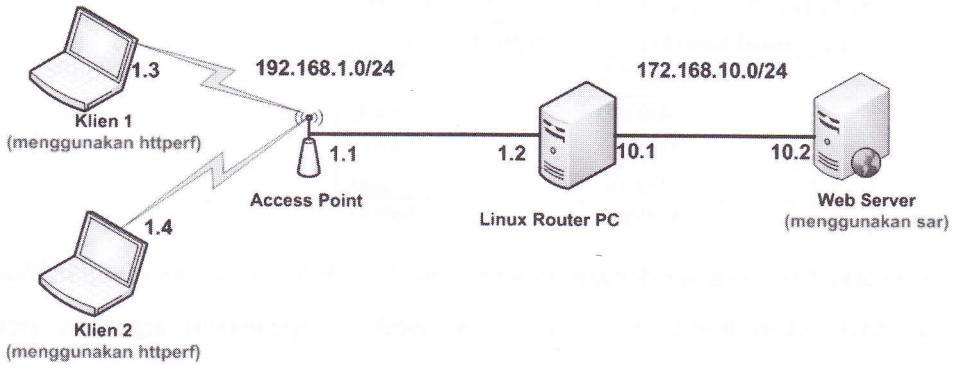
Linux Virtual server dapat diimplementasikan dalam tiga cara. Ada tiga teknik *IP load balancing* dalam sebuah *load balancer*, yaitu :

- Network Address Transrlation (NAT) :
  Suatu metode yang memanipulasi alamat IP dan nomor *port* baik sumber maupun tujuannya. Alamat IP publik disamarkan untuk digunakan oleh alamat IP *private* agar bisa berhubungan dengan dunia luar (internet). Semua proses masuk (*end-user*) dan keluarnya (*real server*) paket harus melalui satu alamat IP publik saja (*director/load balancer/virtual server*).
- Direct Routing :
  Paket dari *end-user* diteruskan secara langsung ke *real server*. IP paket tidak dimodifikasi, jadi *real server* harus dikonfigurasi untuk menerima trafik dari alamat IP *load balancer*.
- IP Tunneling :
  Metode *IP tunneling* melewatkan alamat paket kepada alamat IP untuk diarahkan kepada alamat yang lain, hal ini mungkin dilakukan pada jaringan yang berbeda. Pada proses pemilihan di layer 4 cara ini hampir sama dengan *Direct Routing*, hanya saja ketika paket diteruskan paket tersebut dibungkus pada paket IP kemudian memanipulasi *frame ethernet*.

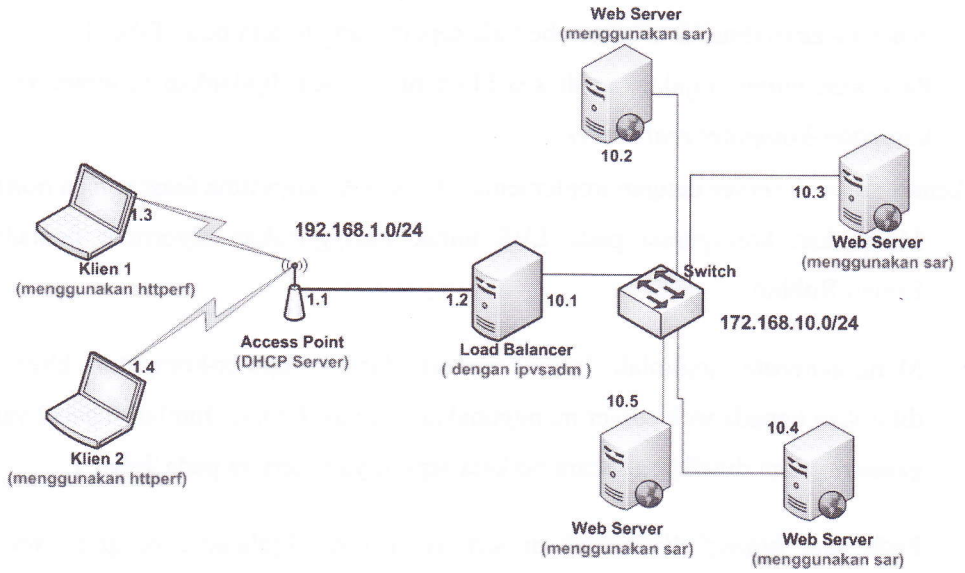
3. TOPOLOGI PENELITIAN

3.1. Topologi Penelitian.

Pada penelitian ini digunakan 2 bentuk topologi masing – masing untuk mengukur performa web server tunggal (Gambar 1) dan web server dengan implementasi load balancing LVS-NAT (Gambar 2).



Gambar 1. Topologi Pengujian Web Server Tunggal.



Gambar 2. Topologi Pengujian Web Server dengan Implementasi LVS.

3.2. Skenario Pengambilan Sampel.

Untuk mengukur kinerja dari web server tanpa implementasi load balancing dan kinerja web server dengan implementasi load balancing maka akan dilakukan beberapa skenario pengujian dengan menggunakan aplikasi-aplikasi pendukung.

1. Skenario 1 (*web server* tunggal tanpa *loab balancing*).
   a. Meng-*generate* sejumlah besar *request* dari komputer-komputer klien yang ditujukan kepada web server menggunakan aplikasi *httperf.* Jumlah *request* yang di-*generate* akan dinaikkan secara berkala seperti yang tertera pada tabel 1.
   b. Pada saat *httperf* dijalankan di sisi klien maka dijalankan juga program *sar* pada komputer-komputer *real server.*

Tabel 1.
Rancangan jumlah beban request pada *httperf.*

No.	Total koneksi	Request/second
1	60000	1000
2	300000	5000
3	600000	10000
4	1200000	20000
5	1800000	30000

2. Skenario 2 (*web server* dengan implementasi LVS-NAT algoritma *round robbin*).
   a. Melakukan konfigurasi pada LVS untuk menggunakan algoritma penjadwalan Round Robbin.
   b. Meng-*generate* sejumlah besar *request* dari komputer-komputer klien yang ditujukan kepada web server menggunakan aplikasi *httperf.* Jumlah *request* yang di-*generate* akan dinaikkan secara berkala seperti yang tertera pada Tabel 1.
   c. Pada saat *httperf* dijalankan di sisi klien maka juga dijalankan program *sar* pada komputer-komputer *real server.*
3. Skenario 3 (*web server* dengan implementasi LVS-NAT algoritma *least connection*).
   a. Melakukan konfigurasi pada LVS untuk menggunakan algoritma penjadwalan Round Robbin.
   b. Meng-*generate* sejumlah besar *request* dari komputer-komputer klien yang ditujukan kepada web server menggunakan aplikasi *httperf.* Jumlah *request* yang di-*generate* akan dinaikkan secara berkala seperti yang tertera pada Tabel 1.
   c. Pada saat *httperf* dijalankan di sisi klien juga dijalankan program *sar* pada komputer-komputer *real server.*

4. PENGAMATAN

Pengamatan dalam penelitian ini dilakukan dalam 3 parameter yaitu *throughput, response time,* dan *CPU Utilization.* Pengamatan dilakukan pada 2 sisi untuk ketiga parameter tersebut yaitu pada sisi klien melalui *tool httperf* untuk melihat *throughput* dan *response time* dan pada sisi *real server* melalui *sar* untuk melihat utilitas CPU.

4.1. Analisis Hasil Pengamatan,

4.1.1 Analisis Hasil Pengamatan Parameter Throughput.

Pengamatan parameter *throughput* dilakukan di sisi klien menggunakan *httperf.* Nilai *throughput* pada penelitian ini mewakili jumlah *request* klien yang dapat direspon oleh *web server* dalam satu satuan waktu. Semakin besar nilai *throughput* yang dihasilkan menunjukkan bahwa kinerja *web server* tersebut semakin baik karena semakin banyak

request klien yang dapat direspon oleh suatu *web server* dalam satu satuan waktu. Hasilnya dapat dilihat pada tabel 2.

Tabel 2.
Perbandingan Throughput (Replies/Second).

Jumlah Koneksi (Request Rate)	server tunggal	Web Server Dengan Implementasi Load balancing LVS-NAT		
		2 server rr	2 server lc	4 server rr
60.000 (1.000request/second)	41.5	49.5	67.7	62
300.000 (5.000request/second)	31.8	60.1	39.5	32.9
600.000 (10.000request/second)	37.1	59.3	42.8	39.9
1.200.000 (20.000 request/second)	35.3	44.9	44.9	149.9
1.800.000 (30.000 request/second)	14.9	55.1	30.5	61.7
Rata-rata	32.12	53.78	45.08	69.28

Keterangan :
- 2 server rr : LVS-NAT algoritma *round robbin* dengan 2 *real server*.
- 2 server lc : LVS-NAT algoritma *least connection* dengan 2 *real server*.
- 4 server rr : LVS-NAT algoritma *round robbin* dengan 4 *real server*.
- 4 server lc : LVS-NAT algoritma *least connection* dengan 4 *real server*.

Hasil Analisis :

a. Web server tunggal hanya mampu menghasilkan rata-rata 32,12 *replies/second*. Nilai ini lebih kecil dibandingkan dengan *web server* yang telah menerapkan *load balancing* menggunakan LVS-NAT yaitu LVS-NAT 2 *server* algoritma *least connection* 45,08 *replies/second*, LVS-NAT 4 *server algoritma least connection* 47,96 *replies/second*, LVS-NAT 2 *server round robbin* 53,78 *replies/second*, dan LVS-NAT 4 *server round robbin* 69,28 *replies/second*. Hal ini menunjukkan bahwa *throughput web server* dengan implementasi *load balancing* menggunakan LVS-NAT lebih baik dibandingkan *throughput web server* tunggal.

b. Web server dengan implementasi load balancing LVS-NAT menggunakan algoritma *round robbin* menunjukkan *throughput* yang lebih baik dibandingkan dengan LVS-NAT algoritma *least connection*. Hal ini dapat dilihat baik pada implementasi *load balancing* dengan 2 *server* maupun 4 *server*. Nilai throughput LVS-NAT *least connection* 2 *server* 45,08 *replies/second* lebih kecil dari nilai *throughput* LVS-NAT *round robbin* 2 *server* 53,78 *replies/second*.

Nilai *throughput* LVS-NAT *least connection* 4 *server* 47,96 *replies/second* lebih kecil dari nilai *throughput* LVS-NAT *round robbin* 4 *server* 69,28 *replies/second.*

c. Nilai *throughput web server* dengan implementasi *load balancing* LVS-NAT 4 server 69,28 *replies/second* merupakan nilai throughput terbaik.

4.1.2. Analisis Hasil Pengamatan Parameter Response Time.

Pengamatan parameter *response time* dilakukan di sisi klien menggunakan *httperf.* Nilai *response time* pada penelitian ini mewakili kecepatan *web server* dalam menanggapi *request* klien. Semakin kecil nilai *response time* yang dihasilkan menunjukkan semakin cepatnya *web server* menanggapi *request* dari klien. Hasilnya dapat dilihat pada tabel 3.

Tabel 3.
Perbandingan Response Time (microsecond)

Jumlah Koneksi (Request Rate)	server tunggal	Web Server Dengan Implementasi Load balancing LVS-NAT		
		2 server rr	2 server lc	4 server rr
60.000 (1.000 request/second)	2089.2	2436.2	2257.6	2141.5
300.000 (5.000 request/second)	2633.5	2286.1	2507.8	2649.4
600.000 (10.000 request/second)	2361.4	2147.4	2480.9	2789.1
1.200.000 (20.000 request/second)	2119.3	2265.2	2545.5	1519.8
1.800.000 (30.000 request/second)	3628.4	2354.5	2827.6	2364.8
Rata-rata	2566.36	2297.88	2523.88	2292.92

Keterangan :
- 2 server rr : LVS-NAT algoritma *round robbin* dengan 2 *real server.*
- 2 server lc : LVS-NAT algoritma *least connection* dengan 2 *real server.*
- 4 server rr : LVS-NAT algoritma *round robbin* dengan 4 *real server.*
- 4 server lc : LVS-NAT algoritma *least connection* dengan 4 *real server.*

Hasil Analisis :

a. Nilai *response time web server* tunggal 2566,36 ms merupakan yang terbesar dibandingkan *web server* LVS-NAT 4 *server least connection* 2528,72 ms, *web server* LVS-NAT 2 *server least connection* 2523,88 ms, *web server* LVS-NAT *round robbin* 2 *server* 2297,88 ms, dan *web server* LVS-NAT *round robbin* 4 *server* 2292,92 ms. Hal ini menunjukkan bahwa penerapan *load balancing* menggunakan LVS-NAT bisa meningkatkan kecepatan *web server* menanggapi *request* klien.

b. Nilai response *time web server* dengan implementasi load *balancing* metode LVS-NAT algoritma *round robbin* 2297,88 ms untuk 2 *server* dan 2292,92 ms untuk 4 server memiliki nilai *response time* yang lebih baik dari *web server* dengan implementasi LVS-NAT menggunakan algoritma *least connection* 2523,88 ms untuk 2 *server* dan 2528,72 ms untuk 4 *server*.

4.1.3. Analisis Hasil Pengamatan Parameter CPU Utilization.

Pengamatan parameter *CPU Utilization* dilakukan di sisi server menggunakan *sar*. Nilai *CPU Utilization* diperoleh dari nilai seluruh sumber daya CPU yang ada yaitu 100% dikurangi persentasi sumber daya tersisa pada kolom % idle output sar. Semakin besar nilai hasil pengurangan tersebut maka semakin baik karena semakin optimal server tersebut bekerja dan menggunakan sumber daya yang ada. Hasilnya dapat dilihat pada tabel 4, 5, 6, 7, dan 8.

Tabel 4.
Persentasi CPU Utilization Web Server Tunggal

Jumlah Koneksi (Request Rate)	server tunggal
60.000(1.000 request/second)	94.47
300.000(5.000 request/second)	95.6
600.000(10.000 request/second)	95.11
1.200.000(20.000 request/second)	94.65
1.800.000(30.000 request/second)	95.54
Rata-rata	95.07

Sumber daya CPU yang digunakan = **100% - 95.07 % = 4.93 %**

Tabel 5.
Persentasi CPU Utilization Web Server Dengan LVS-NAT Algoritma RR menggunakan 2 Real Server

Jumlah Koneksi (Request Rate)	server 1	server 2
60.000(1.000 request/second)	95.71	95.87
300.000(5.000 request/second)	95.72	96.05
600.000(10.000 request/second)	95.70	95.85
1.200.000(20.000 request/second)	96.21	96.40
1.800.000(30.000 request/second)	95.56	95.66
Rata-rata	95.78	95.97

Sumber daya CPU yang digunakan pada server 1= **100% - 95.78% = 4.22 %**
Sumber daya CPU yang digunakan pada server 2 = **100% - 95.97% = 4.03%**

Tabel 6.
Persentasi CPU Utilization Web Server Dengan LVS-NATAlgoritma LC menggunakan 2 Real Server

Jumlah Koneksi (Request Rate)	server 1	server 2
60.000(1.000 request/second)	95.61	96.22
300.000(5.000 request/second)	95.87	96.18
600.000(10.000 request/second)	95.46	96.78
1.200.000(20.000 request/second)	96.63	96.36
1.800.000(30.000 request/second)	97.07	96.60
Rata-rata	96.13	96.43

Sumber daya CPU yang digunakan pada server 1 = **100% - 96.13% = 3.87%**
Sumber daya CPU yang digunakan pada server 2 = **100% - 96.43% = 3.57%**

Tabel 7.
Persentasi CPU Utilization Web Server Dengan LVS-NAT Algoritma RR menggunakan 4 Real Server

Jumlah Koneksi (Request Rate)	server 1	server 2	server 3	server 4
60.000 (1.000 request/second)	97.52	96.47	97.04	97.27
300.000 (5.000 request/second)	97.43	96.94	97.15	96.96
600.000 (10.000 request/second)	97.55	96.96	96.65	97.07
1.200.000 (20.000 request/second)	97.37	96.55	96.54	96.71
1.800.000 (30.000 request/second)	97.51	96.60	96.82	97.01
Rata-rata	97.48	96.70	96.84	97.00

Sumber daya CPU yang digunakan pada server 1 = **100% - 97.48% = 2.52%**
Sumber daya CPU yang digunakan pada server 2 = **100% - 96.70% = 3.3%**
Sumber daya CPU yang digunakan pada server 3 = **100% - 96.84% = 3.16%**
Sumber daya CPU yang digunakan pada server 4 = **100% - 97% = 3%**

Tabel 8.
Persentasi CPU Utilization Web Server Dengan LVS-NAT Algoritma LC menggunakan 4 Real Server

Jumlah Koneksi (Request Rate)	server 1	server 2	server 3	server 4
60.000 (1.000 request/second)	97.56	97.70	97.20	97.17

Tabel 8.
Persentasi CPU Utilization Web Server Dengan LVS-NAT Algoritma LC menggunakan 4 Real Server

Jumlah Koneksi (Request Rate)	server 1	server 2	server 3	server 4
300.000 (5.000 request/second)	97.67	97.72	96.51	96.51
600.000 (10.000 request/second)	97.51	97.95	97.51	97.6
1.200.000 (20.000 request/second)	97.37	97.72	97.54	97.77
1.800.000 (30.000 request/second)	97.68	98.45	96.99	96.53
Rata-rata	97.56	97.91	97.15	97.12

Sumber daya CPU yang digunakan pada server 1 = **100% - 97.56 = 2.44%**
Sumber daya CPU yang digunakan pada server 2 = **100% - 97.91 = 2.09%**
Sumber daya CPU yang digunakan pada server 3 = **100% - 97.15% = 2.85%**
Sumber daya CPU yang digunakan pada server 4 = **100% - 97.12% = 2.88%**

Hasil analisis :

a. Web server tunggal paling banyak memakan sumber daya CPU sebesar 4,93% atau hampir 5% sedangkan *web server* yang telah menerapkan *load balancing* menggunakan LVS-NAT hanya menggunakan sumber daya CPU sampai 4 % yaitu pada penerapan LVS-NAT dengan 2 *server* menggunakan algoritma *round robbin*.

b. Web server yang menggunakan implementasi *load balancing* LVS-NAT algoritma *least connection* 4 *server* paling sedikit menggunakan sumber daya CPU yaitu berkisar antara 2%-3%.

5. KESIMPULAN

Setelah melakukan implementasi *load balancing web server* menggunakan Linux Virtual Server dengan metode LVS-NAT, pengujian kinerja *web server* hasil implementasi *load balancing* tersebut, serta membandingkannya dengan kinerja *web server* tunggal maka didapatkan kesimpulan sebagai berikut :

a) Implementasi *load balancing web server* menggunakan LVS-NAT mampu meningkatkan nilai *throughput web server* dengan besaran yang cukup signifikan hingga 2 kali lipat *throughput web server* tunggal. Nilai ini diperoleh pada implementasi LVS dengan 4 *real server* menggunakan algoritma *round robbin* dengan throughput sebesar 69,28 *replies/second* . Peningkatan *throughput web server* sejalan dengan peningkatan jumlah *real server* yang digunakan dalam LVS. Throughput *web server* dengan LVS 2 *real server* baik itu yang menggunakan algoritma *Round Robbin*

53,78 *replies.second* maupun *Least Connection* 45,08 *replies/second* lebih kecil dari *throughput web server* dengan LVS 4 *real server* menggunakan algoritma *Round Robbin* 69,28 *replies/second* dan *Least Connection* 47,96 *replies/second*.

b) Implementasi *load balancing web server* menggunakan LVS-NAT mampu meningkatkan *response time* dan mengoptimalkan *CPU Utilization* dari *web server* namun peningkatan yang dihasilkan memiliki nilai yang tidak terlalu besar sehingga dianggap tidak terlalu signifikan.

c) Implementasi LVS-NAT menggunakan algoritma *round robbin* lebih handal dalam mengoptimalkan *throughput, CPU Utilization,* dan *response time* dari *web server* jika dibandingkan dengan implementasi LVS-NAT menggunakan algoritma *least connection*.

Daftar Pustaka

Arlitt, M., & Williamson, C. (2004). *Understanding Web Server Configuration Issues*. Canada : Department of Computer and Science, University of Calgary.

Bourke, T. (2001). *Server Load Balancing*. O'Reilly & Associates, Inc., Sebastopol.

Gourley, D., Totty, B., Sayer, M., Sailu, R., & Aggarwal, A. (2002). *HTTP – The Definitive Guide*. O'Reilly.

Kim, M., Choi, M., & Hong, J., *Highly Available and Efficient Load Cluster Management System using SNMP and Web*. Dept. of Computer Science andEngineering, POSTECH Pohang, Korea.

Utdirartatmo, F. (2004). *Clustering PC di Linux dengan OpenMosix dan ClusterKnoppix*, Yogyakarta: Penerbit ANDI.

Tanenbaum, S. (2003). *Computer Networks* – Fourth Edition. Prentice Hall.

Zhang, W. (2004). *Linux Virtual Server for Scalable Network Service, http://www.linuxvirtualserver.org/*, diakses tanggal 16 April 2011

Zhang, W., Jin, S., & Wu, Q.,.(2004). *Creating Linux Virtual Servers*. Hunan-China: National Laboratory for Parallel and Distributed Processing.